# **CS530: Advanced Game Engine Architecture**

# Summer 2012

**Prerequisites:** CS529 & CS541

**Schedule:** Monday & Wednesday 5:00pm—6:40pm

**Classroom:** Carr

**Professors:** Antoine Abi Chakra, Chris Peters

**Contact:** [achacra@digipen.edu](mailto:achacra@digipen.edu)

[cpeters@digipen.edu](mailto:cpeters@digipen.edu)

**Class Web Page:** The **CS530** course at [distance.digipen.edu](http://distance.digipen.edu)

**Office Hours:** Tuesday 3:00pm—6:00pm

**Description**

A game engine is a complex framework or library that provides vital functionalities to any video game independent of the game content or genre. A well-designed game engine must at least provide the following functionalities: data management, rendering, networking, dynamics, input controllers, audio, editing tools, modeling tools, and a high-level application programming interface (API) for the entire framework that hides the low-level details of graphics, networking, and audio programming. Thus, a game engine is a complex library consisting of various components that must all be efficiently integrated into a single framework using the principles of object-oriented design. In this course, students will study the computer graphics, mathematics, data structures, and algorithms required to design and architect a game engine that can handle complex graphics applications that handle three-dimensional data such as games and computer-aided design.

**Course Objectives and Learning Outcomes**

After completing this course, students will have a much deeper understanding of what goes into building modern game engines. They will be familiar with game the major components of game engines such as Graphics, Physics, Sound, Object Systems, Animation, Editor and Content tools. They will have a deep understanding of the interaction of these systems in a large scale engine. For the practical section of the class they will build and use several advanced engine technologies such as reflection and code self inspection. They will have modified and improved a game engine framework.

**Textbooks**

There are no required books for this class.

**Optional Textbooks**

Game Engine Architecture, *by Jason Gregory*, AK Peters (ISBN: 1568814135)

**References**

The optional book listed is the best resource for game engine architecture, but there are many more books, websites, and other resources that can be extremely useful, such as Game Programming Gems. You can find the current list of recommended resources on the **GameCentral** page.

**Outline and Tentative Dates**

This class will roughly follow the outline below, although the order and/or content of the lectures are subject to change.

**Week 1**

1. Introduction
   1. What is a Game Engine?
   2. State of the Industry
   3. The Engine Problem
   4. Overview of Current Engines
   5. Principles of Game Architecture
2. Major Components of Game Engines
   1. Layered Architecture
   2. Graphics
   3. Physics
   4. Sound
   5. Object System
   6. Resource System
   7. Editor / Content Tools

**Week 2**

1. Physical Structure
   1. Headers / Declarations / Precompiled Headers
   2. Compilation Models and Templates
   3. Private Implementation Pattern
   4. Virtual Interface Pattern
   5. Layers , Folders, Projects
   6. External Libraries
   7. DLLs and Modules
   8. Cross Platform Issues
2. Diagnostics and Errors
   1. Assertions
   2. Exceptions
   3. Error Codes
   4. Error Context
   5. Threading
   6. Crashing

**Week 3**

1. Runtime Structure
   1. Memory
      1. The Heap
      2. Block and Pool Allocators
      3. Memory Graphs and Tracking
      4. Memory Leaks and Corruption
      5. Garbage Collection and Handles
      6. Cache and Performance
   2. Logical
      1. Class Layout
      2. Virtual functions and branching
      3. Basic Data Structures
      4. Intrusive Data Structures
      5. Delegates and Callbacks
   3. Debugging Memory
      1. Stack Tracks
      2. Memory Guards

**Week 4**

1. Object System
   1. Object Oriented Design
      1. Properties and Reflection
      2. Monolithic Object Issue
   2. Component Based Design
   3. Property Based Design
   4. Functional Design
   5. Communication
      1. Events
      2. Event Forwarding and Bubbling
      3. Thread Communication
   6. Serialization / Persistence
      1. Basic Data Serialization
      2. Properties
      3. Versioning
      4. Archetypes
      5. Relational Databases
      6. References
   7. References
      1. Object Ownership
      2. Handles
      3. Garbage Collection
2. MetaTypes and Reflection
   1. Reflection
   2. Property Grid
   3. Scripting
   4. Binding
   5. Dynamic Reload

**Week 5**

1. Resource System
   1. Resource Processing
   2. Resource Editing
   3. Referential Integrity
2. Shared Data Management
   1. Large Scene Editing
   2. Version Control
   3. Data Base Systems
   4. Conflict Resolution

**Week 6**

1. Graphics
   1. Modern Rendering
   2. Shaders and Compositing
   3. Basic Forward Lighting
   4. Deferred Lighting
   5. Global Illumination

**Week 7**

1. Game Physics
   1. The Problem with Physics
   2. Collision Detection
   3. Dynamic and Collision Resolution
   4. Constraints and Joints
   5. Physics Events
   6. Precision and Floating Point
   7. Ray Casting

**Week 8**

1. Animation
   1. Skeletal Animation
   2. Property Animation
   3. Animation Blending
   4. Animation and State Machines

**Week 9**

1. Threading Low Level
   1. Threading Primitives
   2. Atomic Data Structures
   3. Lockless Programming
2. High Level Threading
   1. System Threading
   2. Job Systems
   3. Thread Communication

**Week 10**

1. Advanced Game Tools
   1. Perf HUD
   2. Sampling Profiling
   3. Static Code Analysis
   4. Runtime Code Cnalysis (ValGrind, etc)

**Week 11**

**Review.**

**Grading Policy**

The grade for this class will be determinate by the average of the 4 major projects in this class. Each project is grade on a point system. Each assignment has a base starting percentage. Missing or poorly done features will result in negative penalties. Additional and well done features will add to the score. To receive a high grade all required and basic features must be completed and selected additional features must be added.

**Projects**

The class will consist of 3 major projects covering major areas of application development. It is assumed that the students will ether modify the existing game engine sample or using an existing engine of their own.

**Project One – Game Engine Review**

Using an existing game engine created by the student earlier or the sample game engine provided on GameCentral. Write small brief summary of the architecture of the game engine. Topics to cover are the Object Model, Communication Model, and the Serialization Model. These should be the detail level required by another developer to understand how to work in the game engine. This engine will be used for the rest of the assignments in the class.

**Project Two - Dynamic Code Reflection**

Students will implement a system to add dynamic code binding features to C++. This includes exposing properties, methods, and elements to a dynamic interface such as a real time property viewer, scripting language or remote system. This can be done with templates, compiler support, or custom generators.

**Project Three - Memory Analyzer and Optimization**

Students will implement a memory profiling and analysis tool to determinate the nature of dynamic memory allocation in a project. They will then write several memory allocation patterns such as the block allocator and the pool allocator to optimize the memory usage in the project reducing usage and improving speed.

**Project Four - Multithreaded communication and Job System**

Student will implement a multithread communication system for worker threads and tasks. These tasks will be created from the main application and work in the background to communicate state back to the main thread.

**Mechanisms and Procedures**

There are a variety of procedures and mechanisms used in this class to make it run as smoothly as possible. Make sure you read each of these sections thoroughly so that you understand what the instructors expect.

**Late Policy**

If an assignment is turned in late, its grade is reduced by 5% for each day (including weekends) it is late, but never below 0%. After one week, assignments can no longer be turned in.

**Last Day to Withdraw**

In order to withdraw from a course it is not sufficient simply to stop attending class or to inform the instructor. In accordance with the policy, contact your advisor or the registrar to begin the withdrawal process. The last day for withdrawal from this course is cited in the official catalog.

**Academic Integrity Policy**

Cheating, or academic dishonesty in any form, will not be tolerated in this course. Penalties for cheating may include receiving a zero on an assignment, or a failing grade in the course, or even expulsion from DigiPen. For further details, please consult the *DigiPen Academic Integrity Policy*. Note that in this team project class, working directly with your teammates, or even with other teams, is not cheating (and is highly encouraged). However, each student is required to accurately inform the instructors of the exact work they personally did on the project—any deception is cheating and will be punished harshly.

**Disabled Student Services**

Students with physical, psychological or learning disabilities that affect their ability to perform major life activities associated with this class may be eligible for reasonable accommodations under the *Americans with Disabilities Act*. If you have a documented disability please contact the Disability Support Services office to arrange for accommodations for this class.